REPORT

Mini Project: Web Scraping Car Details from Cars24.com

CARS 24

Submitted by

TEAM C

Rohit Sharma	Shameer sutar
Aishwarya Vijay Ighepatil	Ankur
Jigyasa Karakoti	Ramanathan
SANATH KUMAR P S	Girijanath P
Vishal Terence Vaz	Muskan Tripathi
Gopinath .R	Haridhar K B
Sandeep Sharma	Kavidurga vaiyapuri
Ravi Chandra	R.YASWITHA
Varshitha. N	Mandadi Harshitha Reddy
Rahul L Jain	Srinidhi Madanagopal

Project Objective: The objective of this mini-project is to develop skills in web scraping by extracting and analyzing car details from Cars24.com. All team members gathered data specific to the Mumbai location, covering key information such as kilometers driven, year of manufacture, fuel type, transmission, and price. By the end of this project, members will have hands-on experience in web scraping, data cleaning, and presenting structured data for analysis.

METHODOLOGY:

Assigned Brands: Maruti, Tata, Hyundai, Toyota

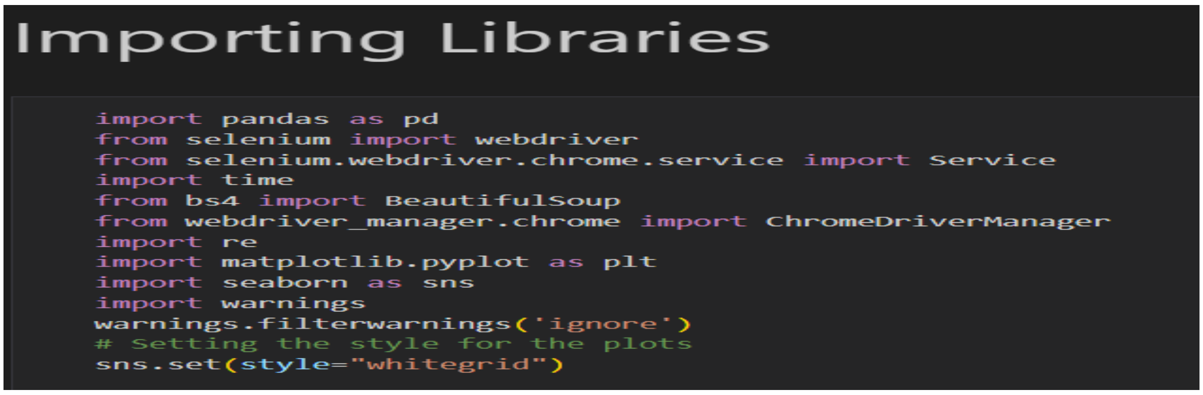
Assigned Location: Mumbai

1) **Research and Planning**:

- Understand the structure of Cars24.com.
- Identify the HTML elements containing the required information.

2) **Data Extraction**: Write a script to scrape the necessary details (kilometers driven, year of manufacture, fuel type, transmission, and price) for the assigned brand.

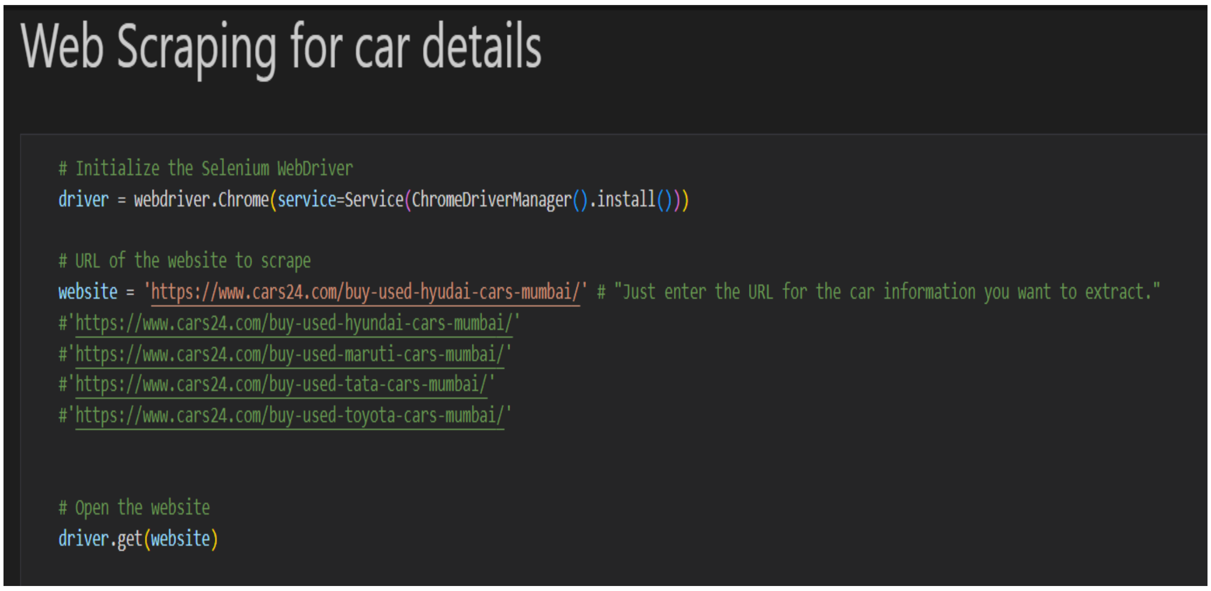
- **Initialize Selenium WebDriver:** Selenium WebDriver is set up and configured to use the Chrome browser. This allows for automated interaction with the website.



Importing Libraries

```
import pandas as pd
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
import time
from bs4 import BeautifulSoup
from webdriver_manager.chrome import ChromeDriverManager
import re
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
# Setting the style for the plots
sns.set(style="whitegrid")
```

- **Open the Target Website:** The WebDriver navigates to the specified URL of the car listing page.


Web Scraping for car details

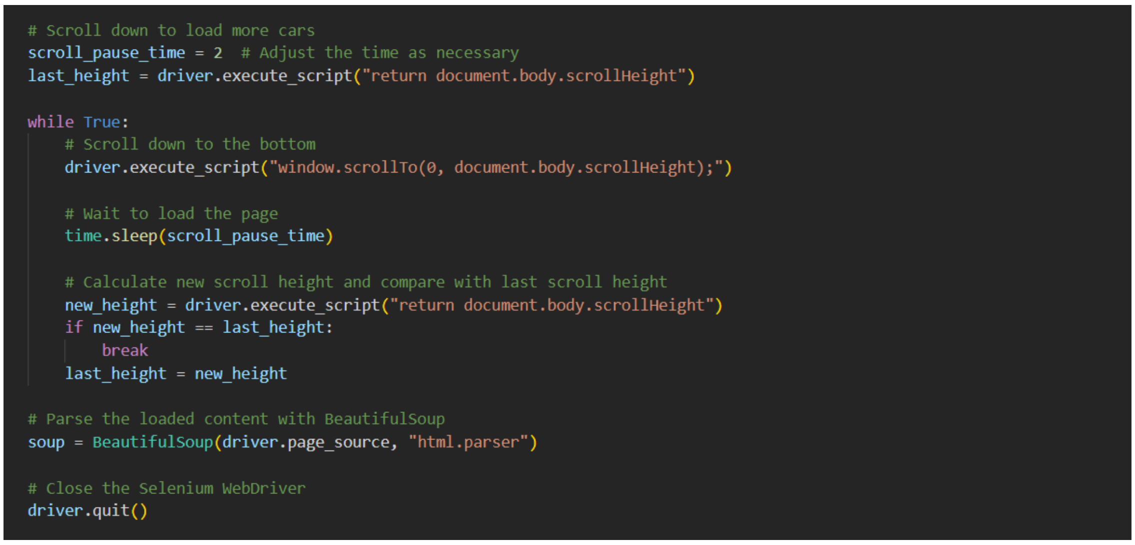
```
# Initialize the Selenium WebDriver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

# URL of the website to scrape
website = 'https://www.cars24.com/buy-used-hyudai-cars-mumbai/' # "Just enter the URL for the car information you want to extract."
#'https://www.cars24.com/buy-used-hyundai-cars-mumbai/'
#'https://www.cars24.com/buy-used-maruti-cars-mumbai/'
#'https://www.cars24.com/buy-used-tata-cars-mumbai/'
#'https://www.cars24.com/buy-used-toyota-cars-mumbai/'


# Open the website
driver.get(website)
```

- **Automating Scrolling:** Selenium allows the script to simulate user actions such as scrolling down the page. This is essential for loading more car listings that are not visible initially.

- **Automate Scrolling to Load All Content:** The script repeatedly scrolls down the page to trigger dynamic loading of additional car listings. It waits for a short period after each scroll to ensure new content is loaded.

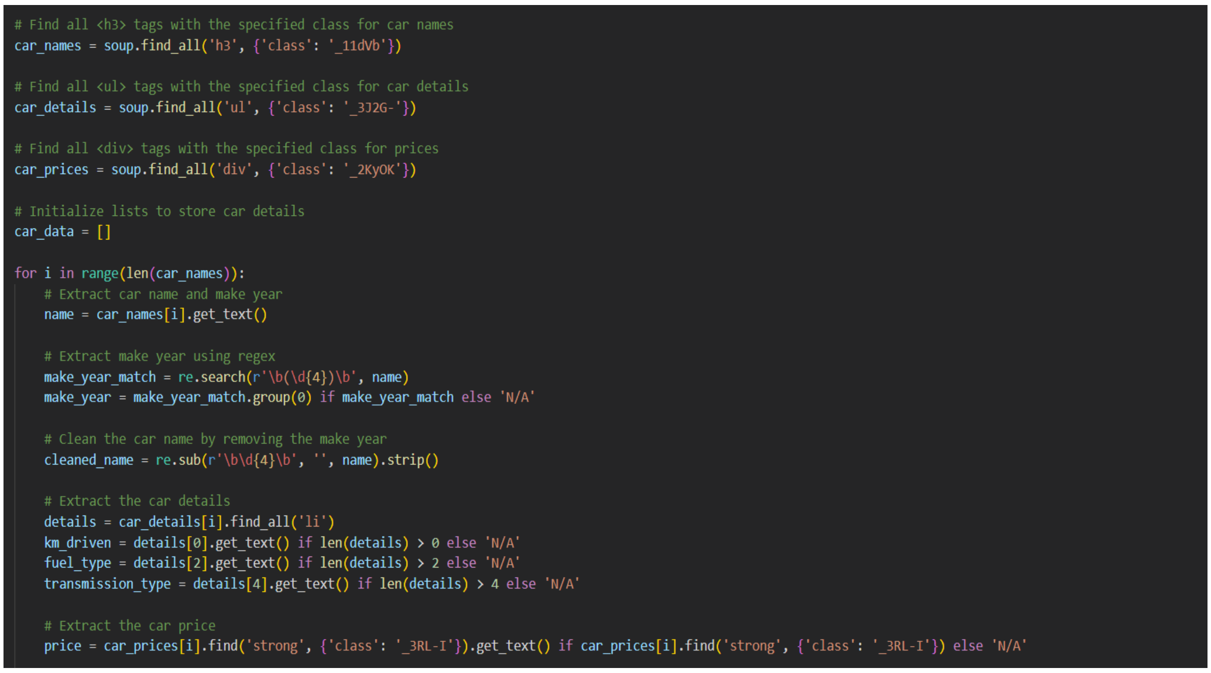
- **Dynamic Content Loading:** Many modern websites, including car listing sites, use JavaScript to load content dynamically. This means that not all the data is present in the initial HTML source but is loaded as the user scrolls or interacts with the page.

- **Parse Loaded Content:** After all content is loaded, the page source is parsed using BeautifulSoup to extract the relevant HTML elements.



```python
# Scroll down to load more cars
scroll_pause_time = 2  # Adjust the time as necessary
last_height = driver.execute_script("return document.body.scrollHeight")

while True:
    # Scroll down to the bottom
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")

    # Wait to load the page
    time.sleep(scroll_pause_time)

    # Calculate new scroll height and compare with last scroll height
    new_height = driver.execute_script("return document.body.scrollHeight")
    if new_height == last_height:
        break
    last_height = new_height

# Parse the loaded content with BeautifulSoup
soup = BeautifulSoup(driver.page_source, "html.parser")

# Close the Selenium WebDriver
driver.quit()
```

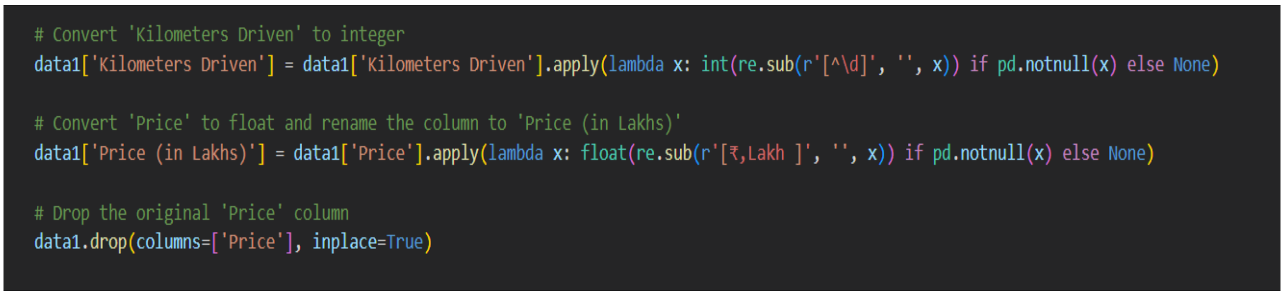
- **<u>Extract Car Details:</u>** The script finds and extracts car names, details, and prices from the HTML elements.
  <u>For each car, it extracts the car name and manufacturing year, kilometers driven, fuel type, and transmission type from the details list and price from the price tag.</u>



```python
# Find all <h3> tags with the specified class for car names
car_names = soup.find_all('h3', {'class': '_11dVb'})

# Find all <ul> tags with the specified class for car details
car_details = soup.find_all('ul', {'class': '_3J2G-'})

# Find all <div> tags with the specified class for prices
car_prices = soup.find_all('div', {'class': '_2KyOK'})

# Initialize lists to store car details
car_data = []

for i in range(len(car_names)):
    # Extract car name and make year
    name = car_names[i].get_text()

    # Extract make year using regex
    make_year_match = re.search(r'\b(\d{4})\b', name)
    make_year = make_year_match.group(0) if make_year_match else 'N/A'

    # Clean the car name by removing the make year
    cleaned_name = re.sub(r'\b\d{4}\b', '', name).strip()

    # Extract the car details
    details = car_details[i].find_all('li')
    km_driven = details[0].get_text() if len(details) > 0 else 'N/A'
    fuel_type = details[2].get_text() if len(details) > 2 else 'N/A'
    transmission_type = details[4].get_text() if len(details) > 4 else 'N/A'

    # Extract the car price
    price = car_prices[i].find('strong', {'class': '_3RL-I'}).get_text() if car_prices[i].find('strong', {'class': '_3RL-I'}) else 'N/A'
```

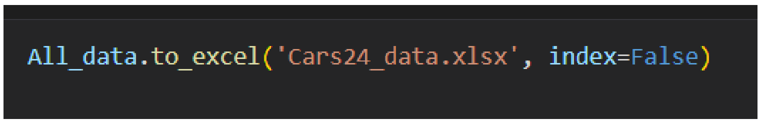
3) **Data Cleaning**: Ensure the scraped data is clean and organized for analysis.

- **Clean and Structure Data:** The extracted data is cleaned (e.g., removing the make year from car names) and structured into a list.



```
# Convert 'Kilometers Driven' to integer
data1['Kilometers Driven'] = data1['Kilometers Driven'].apply(lambda x: int(re.sub(r'[^\d]', '', x)) if pd.notnull(x) else None)

# Convert 'Price' to float and rename the column to 'Price (in Lakhs)'
data1['Price (in Lakhs)'] = data1['Price'].apply(lambda x: float(re.sub(r'[₹,Lakh ]', '', x)) if pd.notnull(x) else None)

# Drop the original 'Price' column
data1.drop(columns=['Price'], inplace=True)
```

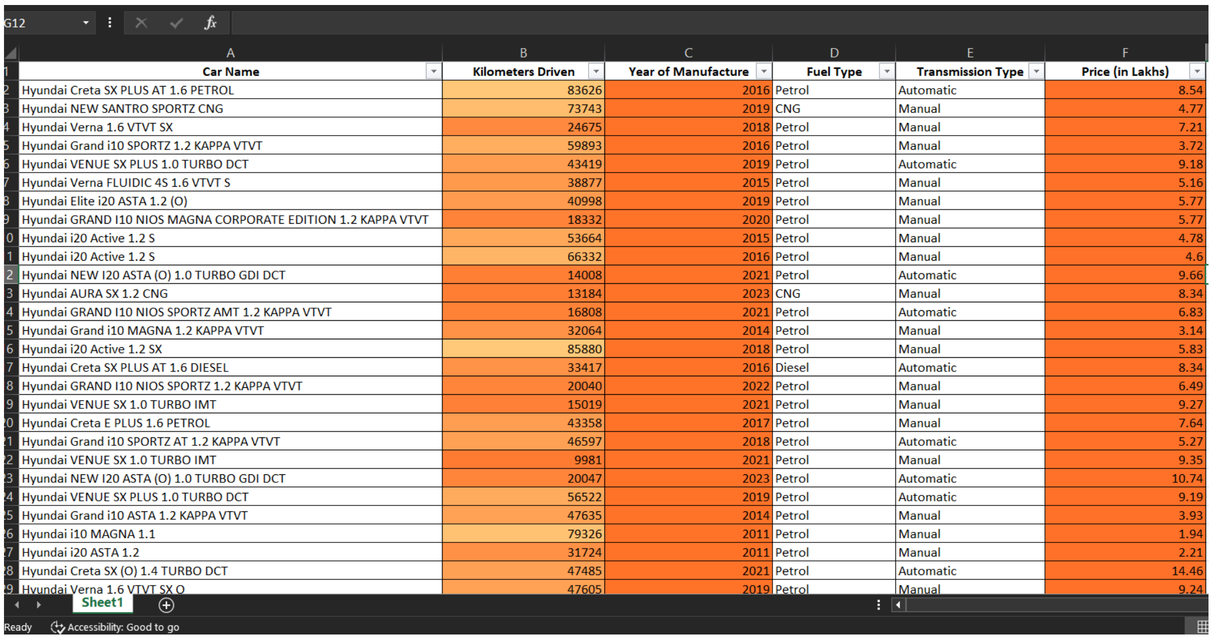
4) Saving Data into file: The structured data is converted into a pandas DataFrame and saved to a CSV file.



```
All_data.to_excel('Cars24_data.xlsx', index=False)
```

5) **Data Presentation**: Present the collected data in a structured format in a CSV.

DATA

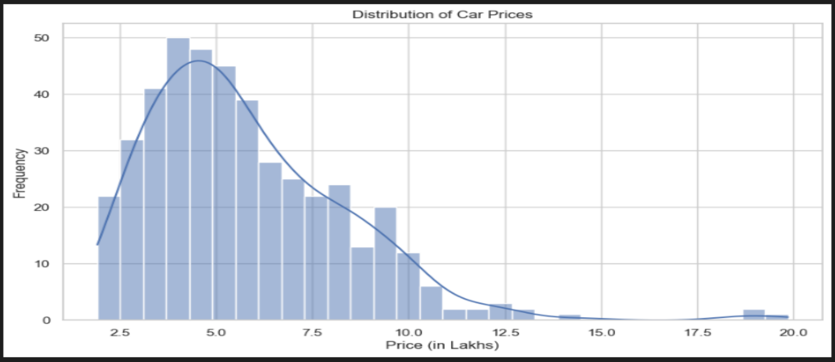


	Car Name	Kilometers Driven	Year of Manufacture	Fuel Type	Transmission Type	Price (in Lakhs)
2	Hyundai Creta SX PLUS AT 1.6 PETROL	83626	2016	Petrol	Automatic	8.54
3	Hyundai NEW SANTRO SPORTZ CNG	73743	2019	CNG	Manual	4.77
4	Hyundai Verna 1.6 VTVT SX	24675	2018	Petrol	Manual	7.21
5	Hyundai Grand i10 SPORTZ 1.2 KAPPA VTVT	59893	2016	Petrol	Manual	3.72
6	Hyundai VENUE SX PLUS 1.0 TURBO DCT	43419	2019	Petrol	Automatic	9.18
7	Hyundai Verna FLUIDIC 4S 1.6 VTVT S	38877	2015	Petrol	Manual	5.16
8	Hyundai Elite i20 ASTA 1.2 (O)	40998	2019	Petrol	Manual	5.77
9	Hyundai GRAND I10 NIOS MAGNA CORPORATE EDITION 1.2 KAPPA VTVT	18332	2020	Petrol	Manual	5.77
10	Hyundai i20 Active 1.2 S	53664	2015	Petrol	Manual	4.78
11	Hyundai i20 Active 1.2 S	66332	2016	Petrol	Manual	4.6
12	Hyundai NEW I20 ASTA (O) 1.0 TURBO GDI DCT	14008	2021	Petrol	Automatic	9.66
13	Hyundai AURA SX 1.2 CNG	13184	2023	CNG	Manual	8.34
14	Hyundai GRAND I10 NIOS SPORTZ AMT 1.2 KAPPA VTVT	16808	2021	Petrol	Automatic	6.83
15	Hyundai Grand i10 MAGNA 1.2 KAPPA VTVT	32064	2014	Petrol	Manual	3.14
16	Hyundai i20 Active 1.2 SX	85880	2018	Petrol	Manual	5.83
17	Hyundai Creta SX PLUS AT 1.6 DIESEL	33417	2016	Diesel	Automatic	8.34
18	Hyundai GRAND I10 NIOS SPORTZ 1.2 KAPPA VTVT	20040	2022	Petrol	Manual	6.49
19	Hyundai VENUE SX 1.0 TURBO IMT	15019	2021	Petrol	Manual	9.27
20	Hyundai Creta E PLUS 1.6 PETROL	43358	2017	Petrol	Manual	7.64
21	Hyundai Grand i10 SPORTZ AT 1.2 KAPPA VTVT	46597	2018	Petrol	Automatic	5.27
22	Hyundai VENUE SX 1.0 TURBO IMT	9981	2021	Petrol	Manual	9.35
23	Hyundai NEW I20 ASTA (O) 1.0 TURBO GDI DCT	20047	2023	Petrol	Automatic	10.74
24	Hyundai VENUE SX PLUS 1.0 TURBO DCT	56522	2019	Petrol	Automatic	9.19
25	Hyundai Grand i10 ASTA 1.2 KAPPA VTVT	47635	2014	Petrol	Manual	3.93
26	Hyundai i10 MAGNA 1.1	79326	2011	Petrol	Manual	1.94
27	Hyundai i20 ASTA 1.2	31724	2011	Petrol	Manual	2.21
28	Hyundai Creta SX (O) 1.4 TURBO DCT	47485	2021	Petrol	Automatic	14.46
29	Hyundai Verna 1.6 VTVT SX O	47605	2019	Petrol	Manual	9.24

6)Data Visualization using Python Libraries:

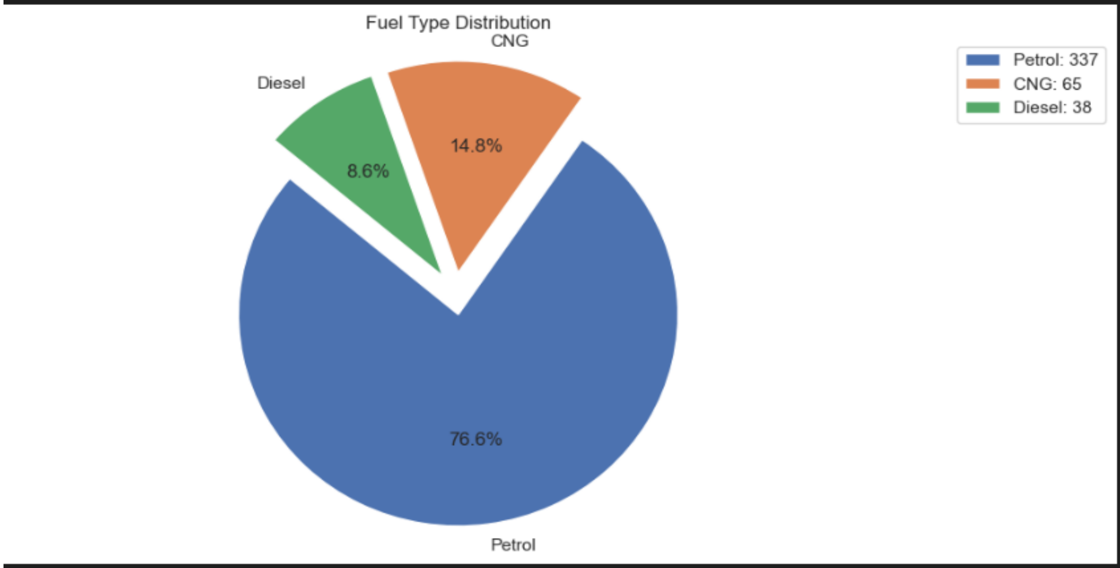
Data Visualization and insights on total number of cars

```python
# 1. Plot the distribution of car prices
plt.figure(figsize=(10, 6))
sns.histplot(df['Price (in Lakhs)'], kde=True, bins=30)
plt.title('Distribution of Car Prices')
plt.xlabel('Price (in Lakhs)')
plt.ylabel('Frequency')
plt.show()
```



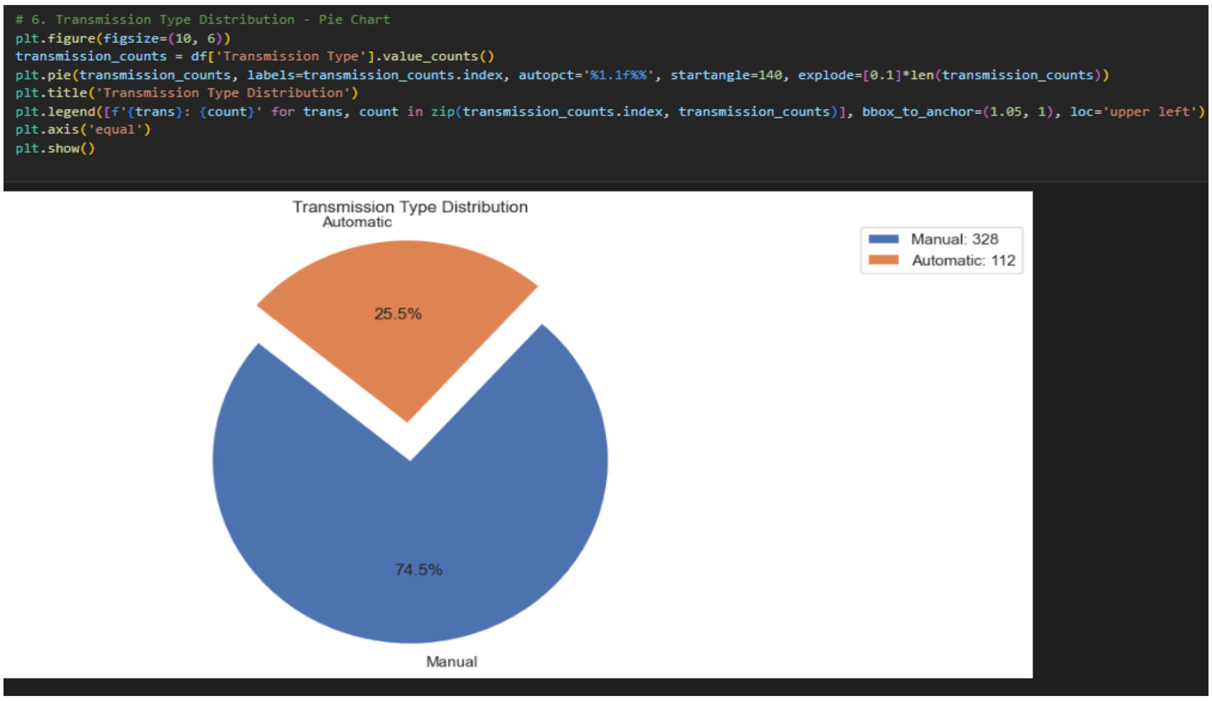
The histogram shows the distribution of car prices across all brands. The distribution appears to be right-skewed, with most cars priced between 2-8 lakhs. There's a long tail extending towards higher prices, indicating some luxury or high-end models. The peak of the distribution is around 4-6 lakhs, suggesting this is the most common price range for used cars in this dataset.

```python
# 5. Fuel Type Distribution - Pie Chart
plt.figure(figsize=(10, 6))
fuel_counts = df['Fuel Type'].value_counts()
plt.pie(fuel_counts, labels=fuel_counts.index, autopct='%1.1f%%', startangle=140, explode=[0.1]*len(fuel_counts))
plt.title('Fuel Type Distribution')
plt.legend([f'{fuel}: {count}' for fuel, count in zip(fuel_counts.index, fuel_counts)], bbox_to_anchor=(1.05, 1), loc='upper left')
plt.axis('equal')
plt.show()
```



The pie chart shows that petrol cars dominate the market, making up about 76.6% of the cars. Diesel cars are the second most common at around 14.8%.

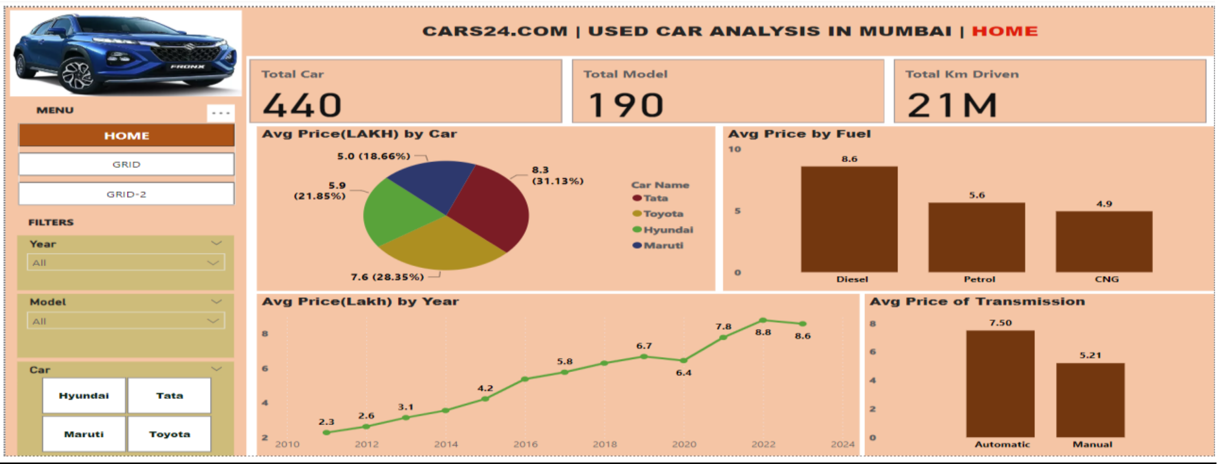
CNG and other fuel types make up a small portion of the market. This distribution likely reflects the general preference for petrol cars in India, especially in urban areas.



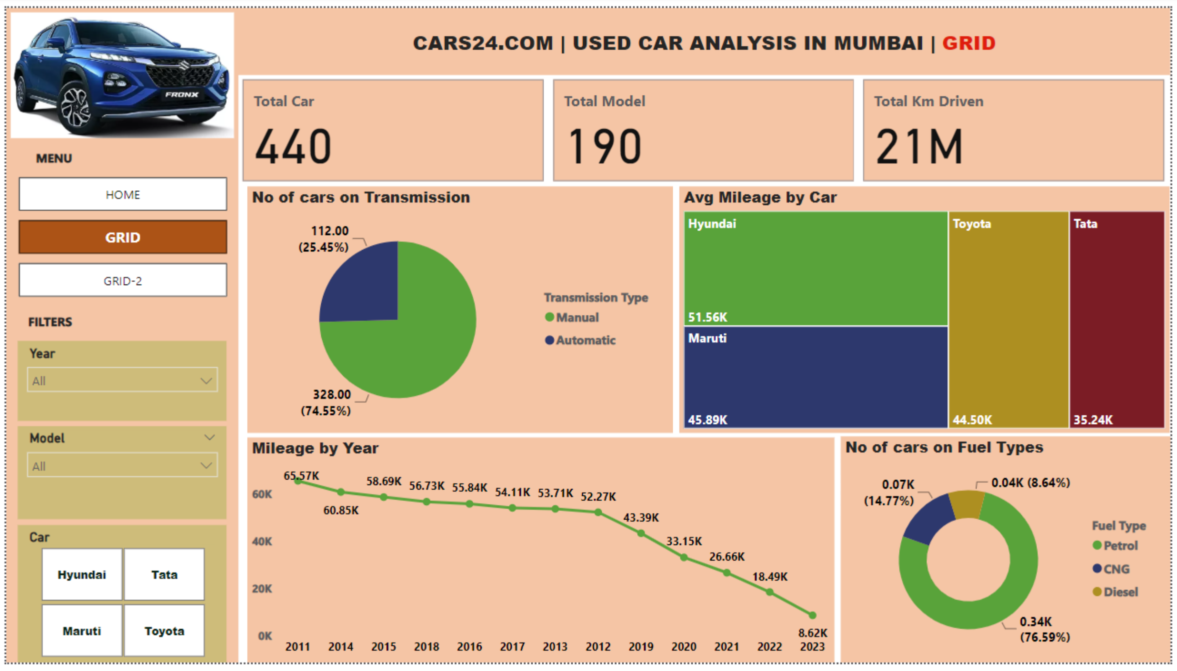
```python
# 6. Transmission Type Distribution - Pie Chart
plt.figure(figsize=(10, 6))
transmission_counts = df['Transmission Type'].value_counts()
plt.pie(transmission_counts, labels=transmission_counts.index, autopct='%1.1f%%', startangle=140, explode=[0.1]*len(transmission_counts))
plt.title('Transmission Type Distribution')
plt.legend([f'{trans}: {count}' for trans, count in zip(transmission_counts.index, transmission_counts)], bbox_to_anchor=(1.05, 1), loc='upper left')
plt.axis('equal')
plt.show()
```

Manual transmission cars are more common, making up about 75% of the cars. Automatic transmission cars account for the remaining 25%. This reflects the Indian market's preference for manual cars, which are generally cheaper and more fuel-efficient.

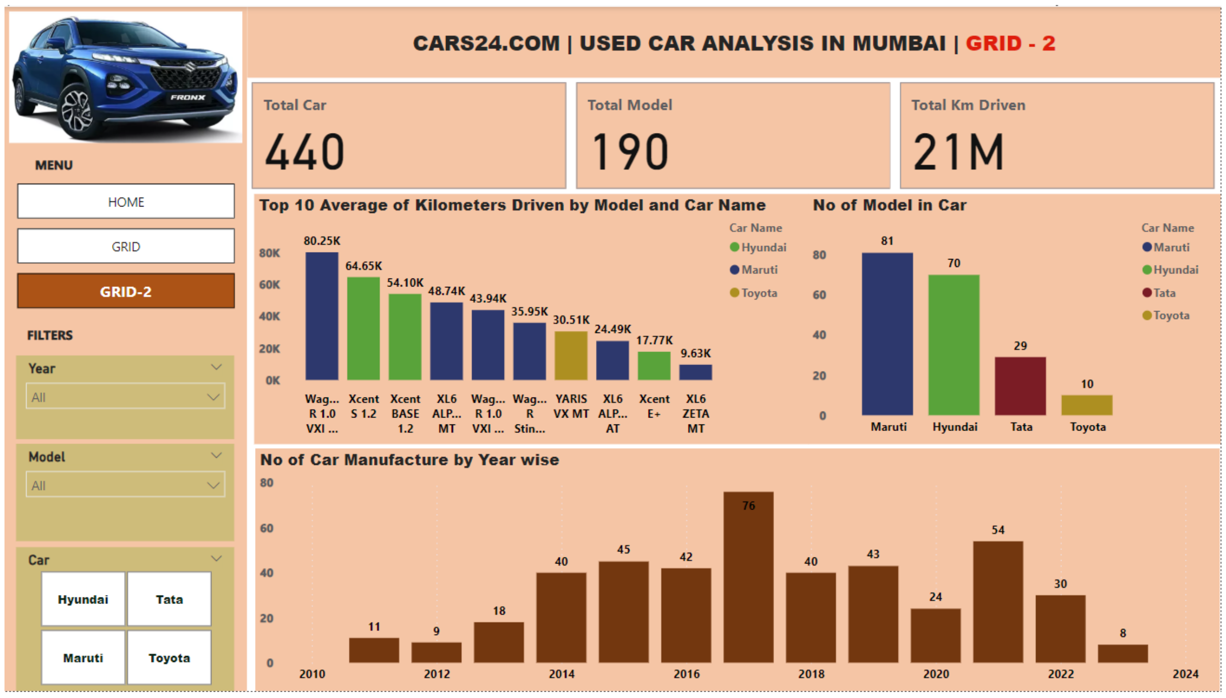
7)Analyzing Gathered Data: Used Microsoft Power BI to create Interactive Dashboards.



<u>Figure(i)-</u> Used Car Analysis In Mumbai



Figure(ii)-_Manual transmission cars are more common



Figure(iii)-Average of Kilometers Driven by a Car

Insights gained from the data:

Price Considerations:

- Most used cars in the market are priced between 2-8 lakhs, with the most common range being 4-6 lakhs.
- Newer cars (2019-2022 models) are more abundant and generally pricier.
- Diesel cars tend to be more expensive, followed by petrol, with CNG being the most affordable option.
- Automatic transmission vehicles command a premium over manual ones.

Mileage Insights:

- The majority of used cars have been driven between 20,000 to 80,000 kilometers.
- Higher mileage generally correlates with lower prices, but other factors like brand and condition also play a role.

Fuel Type Availability:

- Petrol cars dominate the market (about 80%), offering the widest choice.
- Diesel cars make up about 15% of the market and tend to be pricier.
- CNG cars are less common but offer an economical option for budget-conscious buyers.

Transmission Options:

- Manual transmission cars are more widely available (75% of the market) and generally cheaper.
- Automatic cars, while pricier, are increasing in availability, especially in newer models.

Brand-Specific Insights:

Maruti: Offers the most affordable options with a focus on manual transmission and petrol engines.

Tata: Shows rapid price increases in recent models. Offers more diesel options compared to other brands.

Hyundai: Provides a balanced mix of fuel types and steady price progression over the years.

Toyota: Maintains higher prices even for older models. Offers mostly petrol cars with a good mix of automatic options.

Challenges and Overcoming Them:

- **Dynamic Data Loading:** BeautifulSoup alone was inadequate due to dynamic content loading on the website. This challenge was overcome by using Selenium to automate browser actions and fully load the data before extraction.

- **Data Cleaning:** The extracted data required significant cleaning, especially in standardizing car names and separating the manufacture year from the car name for better readability and conversion to correct data types.